**Aim**

# Experiment-1 (A)

To implement logical rules in Python.

## Theory

#### Useful packages to be installed.

**NumPy**

NumPy is a general-purpose array-processing package.

It provides a high-performance multidimensional array object, and tools for working with these arrays.

It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions

#### Pandas

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series.

* Fast and efficient for manipulating and analyzing data.
* Data from different file objects can be loaded.

**Libraries to be included**

#### Kanren

Kanren is a library within PyPi that simplifies ways of making business logic out of code. The logic, rules, and facts we discussed previously can be turned into code using ‘kanren’. It uses advanced forms of pattern matching to understand the input expressions and build its own logic from the given input.

We will be using this library in the sections below for mathematical computations. The import and installation steps are mentioned in the code section that follows.

#### SymPy:

SymPy stands for symbolic computation in Python and is an open-sourced library. It is used for calculating mathematical constructs using symbols.

The aim of the SymPy project is to establish a completely featured Computer Algebra System (CAS). The aim here is to keep the understanding of the code simple and comprehensive.

#### Evaluating mathematical expressions using logic programming

Algorithms are nothing but implementation of logic and control. Similarly, when the logic runs a mathematical function, we call it a mathematical expression. These expressions are the inputs we give to the program, based on which the program understands the rules that are present in the logic.

1

#### Associative Property

The commutative property, in [mathematics,](https://www.britannica.com/science/mathematics) either of two laws relating to number operations of addition and multiplication, stated symbolically: *a* + (*b* + *c*) = (*a* + *b*) + *c*, and *a*(*bc*) = (*ab*)*c*; that is, the terms or factors may be associated in any way desired.

#### Commutative property

The commutative property states that the numbers on which we operate can be moved or swapped from their position without making any difference to the answer. The property holds for Addition and Multiplication, but not for subtraction and division.

## Program

from kanren import run, var, fact

from kanren.assoccomm import eq\_assoccomm as eq from kanren.assoccomm import commutative, associative add = 'add'

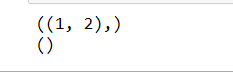
mul = 'mul' fact(commutative, mul) fact(commutative, add) fact(associative, mul) fact(associative, add) a, b = var('a'), var('b')

Original\_pattern = (mul, (add, 5, a), b) exp1 = (mul, 2, (add, 5, 1))

exp2 = (add,5,(mul,8,1))

print(run(0, (a,b), eq(Original\_pattern, exp1))) print(run(0, (a,b), eq(Original\_pattern, exp2)))

## Output:



**Result**

The first experiment has been successfully created and completed.

## Aim

To check for prime numbers.

## Theory

**Kanren**

# Experiment-1 (B)

#### Libraries to be included.

It provides us a way to simplify the way we made code for business logic. It lets us express the logic in terms of rules and facts.

It is a library within PyPi that simplifies ways of making business logic out of code.The logic, rules, and facts we discussed previously can be turned into code using ‘kanren’.

It uses advanced forms of pattern matching to understand the input expressions and build its own logic from the given input.

We will be using this library in the sections below for mathematical computations. The import and installation steps are mentioned in the code section that follows.

#### Sympy

SymPy stands for symbolic computation in Python and is an open-sourced library. It is used for calculating mathematical constructs using symbols.

The aim of the SymPy project is to establish a completely featured Computer Algebra System (CAS). The aim here is to keep the understanding of the code simple and comprehensive.

With the help of logic programming, we can find the prime numbers from a list of numbers and can also generate prime numbers. The Python code given below will find the prime number from a list of numbers and will also generate the first 10 prime numbers.

#### To check for prime numbers

With the help of logic programming, we can find the prime numbers from a list of numbers and can also generate prime numbers. The Python code given below will find the prime number from a list of numbers and will also generate the first 10 prime numbers.

**Program**

from kanren import isvar, run, membero

from kanren.core import goaleval, condeseq, success, fail, eq, var from sympy.ntheory.generate import prime, isprime

import itertools as it

def prime\_check(x):

if isvar(x):

return condeseq([(eq,x,p)] for p in map(prime, it.count(1))) else:

return success if isprime(x) else fail x = var()

print((set(run(0,x,(membero,x,(12,14,15,19,20,21,22,23,29,30,41,44,52,62,65,85)),

(prime\_check,x))))) print((run(10,x,prime\_check(x))))

## Output



**Result**

The experiment has been understood and completed successfully.

# Experiment -2

**Aim**: Using any data apply the concept of: Linear regression.

### Theory:

#### What is linear regression?

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression.

Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

y= a0+a1x+ ε

Y=Dependent Variable (Target Variable) X=Independent Variable (predictor Variable)

a0= intercept of the line (Gives an additional degree of freedom)

a1 = Linear regression coefficient (scale factor to each input value). ε = random error

#### Libraries to be included are:

**Numpy:**

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

#### Why use Numpy ?

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

#### Pandas

Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively.

It provides various data structures and operations for manipulating numerical data and time series.

This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

#### Why use panda?

Fast and efficient for manipulating and analyzing data. Data from different file objects can be loaded.

Easy handling of missing data (represented as NaN) in floating point as well as non -floating point data

Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects.

#### Seaborne

Seaborn is an amazing visualization library for statistical graphics plotting in Python.

It provides beautiful default styles and color palettes to make statistical plots more attractive.

It is built on the top of [matplotlib](https://www.geeksforgeeks.org/python-introduction-matplotlib/) library and also closely integrated to the data structures from [pandas.](https://www.geeksforgeeks.org/introduction-to-pandas-in-python/)

Seaborn aims to make visualization the central part of exploring and understanding data. between different visual representations for same variables for better understanding of dataset.

Why do we need seaborn?

By using the seaborn library, **we can easily represent our data on a plot**. This library is used to visualize our data; we do not need to take care of the internal details; we just have to pass our data set or data inside the relplot() function, and it will calculate and place the value accordingly.

#### Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays.

Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

It was introduced by John Hunter in the year 2002. Why do we need matplotlib?

Matplotlib is a comprehensive library for **creating static, animated, and interactive visualizations in Python**.

Matplotlib makes easy things easy and hard things possible. Create publication quality plots. Make interactive figures that can zoom, pan, update.

As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) **to embed plots in GUI applications.**

**Program**

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

df=pd.read\_csv("/content/Salary\_Data.csv") df.head()

x = df['YearsExperience'] y = df['Salary']

df.head()



def linear\_regression(x, y): N = len(x)

x\_mean = x.mean() y\_mean = y.mean()

B1\_num = ((x - x\_mean) \* (y - y\_mean)).sum() B1\_den = ((x - x\_mean)\*\*2).sum()

B1 = B1\_num / B1\_den

B0 = y\_mean - (B1\*x\_mean)

7

reg\_line = 'y = {} + {}β'.format(B0, round(B1, 3))

return (B0, B1, reg\_line) def corr\_coef(x, y):

N = len(x)

num = (N \* (x\*y).sum()) - (x.sum() \* y.sum())

den = np.sqrt((N \* (x\*\*2).sum() - x.sum()\*\*2) \* (N \* (y\*\*2).sum() - y.sum()\*\*2))

R = num / den return R

Applying these functions to our data, we can print out the results: B0, B1, reg\_line = linear\_regression(x, y)

print('Regression Line: ', reg\_line) R = corr\_coef(x, y) print('Correlation

Coef.: ', R) print('"Goodness of Fit": ', R\*\*2) plt.figure(figsize=(12

,5))

plt.scatter(x, y, s=300, linewidths=1, edgecolor='black') text = '''X Mean: {} Years

Y Mean: ${}

R: {}

R^2: {} y = {} +

{}X'''.format(ro

und(x.mean(), 2),

round(y.mean(),

2), round(R, 4),

round(R\*\*2, 4),

round(B0, 3),

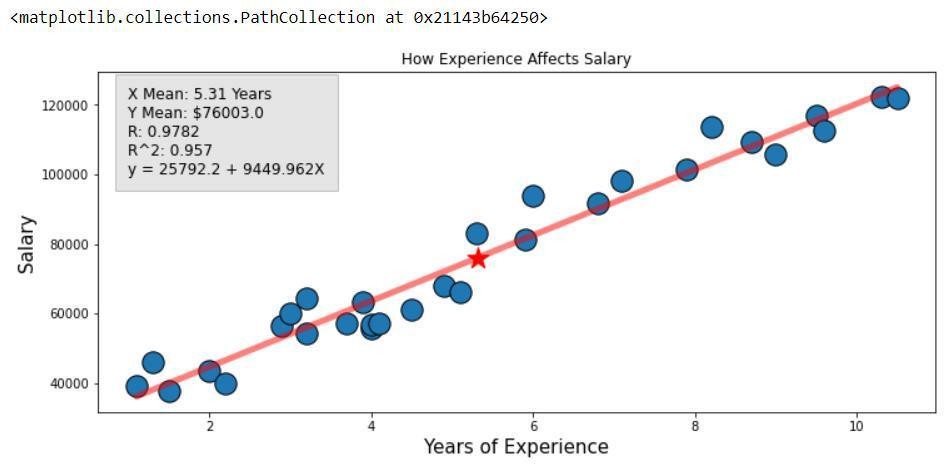
round(B1, 3))

plt.text(x=1, y=100000, s=text, fontsize=12, bbox={'facecolor': 'grey', 'alpha': 0.2, 'pad': 10}) plt.title('How Experience Affects Salary')

plt.xlabel('Years of Experience', fontsize=15) plt.ylabel('Salary', fontsize=15)

plt.plot(x, B0 + B1\*x, c = 'r', linewidth=5, alpha=.5, solid\_capstyle='round') plt.scatter(x=x.mean(), y=y.mean(), marker='\*', s=10\*\*2.5, c='r') # average point

### Output:



**Result:**

The experiment has been completed and understood successfully.